# Real-time Motion Tracking from a Mobile Robot

Boyoon Jung, *Student Member, IEEE,* Gaurav S. Sukhatme, *Member, IEEE*

***Abstract*— A mobile robot needs to perceive the motions of external objects to perform tasks successfully in a dynamic environment. We propose a set of algorithms for multiple motion tracking from a mobile robot equipped with a monocular camera and a laser rangefinder. The key challenges are 1. to compensate the ego-motion of the robot for external motion detection, and 2. to cope with transient and structural noise for robust motion tracking. In our algorithms, the robot ego-motion is directly estimated using corresponding feature sets in two consecutive images, and the position and velocity of a moving object is estimated in image space using multiple particle filters. The estimates are fused with the depth information from the laser rangefinder to estimate the partial 3D position. The proposed algorithms have been tested with various configurations in outdoor environments. The algorithms were deployed on three different platforms; it was shown that various type of ego-motion were successfully eliminated and the particle filter was able to track motions robustly. The multiple target tracking algorithm was tested for different types of motions, and it was shown that our multiple filter approach is effective and robust. The tracking algorithm was integrated with a robot control loop, and its real-time capability was demonstrated.**

***Index Terms*— mobile robot, motion tracking, ego-motion compensation, particle filter.**

## I. INTRODUCTION

MOTION TRACKING is a fundamental capability that a mobile robot must have in order to operate in a dynamic environment. Moving objects (*eg.*, people) are often subjects for a robot to interact with, and in other contexts (*eg.*, traffic) they could be potentially more dangerous for safe navigation compared to stationary objects. Further, capabilities like localization and mapping critically depend on separating moving objects from static ones. Finally motion is the most critical feature to track for many surveillance or security applications. For instance, a building monitoring system can watch for a burglar at night by detecting motion, or an autonomous rescue vehicle can search for victims of natural disaster by sensing motion. Clearly, robust motion detection and tracking are key enablers for many mobile robot applications.

The motion tracking problem from a mobile robot is illustrated in Figure 1. There are multiple moving objects in the vicinity of a mobile robot. Measurements from sensors onboard the robot are contaminated with noise, and an estimation process is required to compute the positions and velocities of the moving objects in the robot's local coordinate system. In the variant of the problem studied here, we require real-time estimates without prior knowledge about the number of moving objects, the motion model of objects, or the structure of the environment. As an additional restriction, a populated, unstructured *outdoor environment* is assumed. Moving object detection in a structured indoor environment has been relatively well studied [1], [2], [3]. In an indoor environment, moving objects are often clearly distinguishable from the rest of the environment due to distinct environmental structures (*eg.* straight and perpendicular walls). In contrast, an outdoor environment contains objects of irregular shapes, and it is challenging to segment moving objects from the background. In addition, outdoor environments contain diverse motions (varying speeds, frequencies etc.).

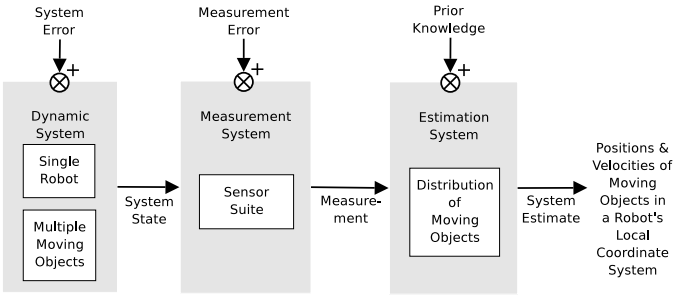



Fig. 1. Motion tracking from a mobile robot: The problem is to estimate the positions and velocities of target motions in the robot's local coordinate system.

There are two main challenges in the motion tracking problem. First, there are two *independent motions* involved - the ego-motion of the mobile robot and the external motions of moving objects. Since these two motions appear blended in the sensor data, the ego-motion of the robot needs to be eliminated so that the remaining motions, which are due to moving objects, can be detected. Second, there are *various types of noise* added at various stages. For example, real outdoor images are contaminated by various noise sources including poor lighting conditions, camera distortion, unstructured and changing shape of objects, etc. Perfect ego-motion compensation is rarely achievable, thus it adds another type of uncertainty to the system. Some of these noise terms are transient and some of them are constant over time.

Our approach to the problem is to design a simple and fast ego-motion compensation algorithm in the pre-processing stage for real-time performance, and to develop a probabilistic filter in the post-processing stage for uncertainty and noise handling. Since the sequence of camera images contains rich information of object motion, a monocular camera is utilized for motion detection and tracking. A laser rangefinder provides depth information of image pixels for partial 3D position estimation. Figure 2 shows the processing sequence of our motion tracking system. Frame differencing, which compares two consecutive images and finds motions based on the difference, is exploited for motion detection. However, when the camera moves (*eg.* when it is mounted on a mobile robot),

Boyoon Jung and Gaurav S. Sukhatme are with the Robotic Embedded Systems Laboratory, Center for Robotics and Embedded Systems, University of Southern California, Los Angeles, USA. E-mail: boyoon,gaurav@robotics.usc.edu.

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **2005** | | **00-00-2005 to 00-00-2005** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **Real-time Motion Tracking from a Mobile Robot** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>**University of Southern California,Computer Science Department,Los Angeles,CA,90089-0781** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES
**The original document contains color images.**

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | |
| **unclassified** | **unclassified** | **unclassified** | | **18** | |

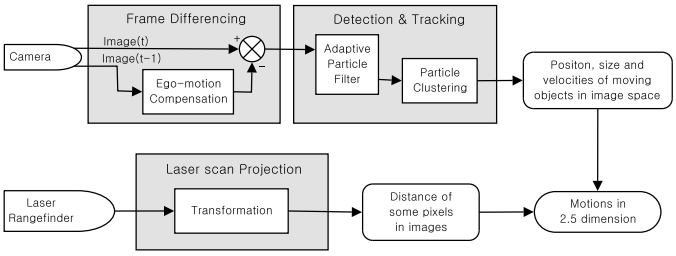



Fig. 2. Processing sequence for motion tracking from a mobile robot: A monocular camera is utilized for motion detection and tracking in image space, and the result is fused with laser scans for motion estimation in 2.5 dimension.

straightforward differencing is not applicable because a big difference is generated by the motion of the camera even when nothing moves in the environment. Therefore, the ego-motion of the camera is compensated before comparing the previous image ($Image(t–1)$) with the current one ($Image(t)$). Assuming that the ego-motion compensation is perfect, the difference image would still contain structured noise on the boundaries of objects because of the lack of depth information from a monocular image. We use a probabilistic model to filter such noise out and to perform robust detection and tracking. The motions of moving objects are modeled using a Bayesian framework, and their probability distribution in image space is estimated by applying perception and motion models. Once the positions and velocities of moving objects are estimated in 2D image space, the information is combined with the partial depth information from a laser rangefinder in order to construct a 3D motion model. By projecting range values into an image space, the image pixels at the same height as the laser rangefinder will have depth information.

The performance of the proposed system has been analyzed in three steps. First, the robustness of the motion tracking algorithms has been tested on various robot platforms that have unique characteristics in terms of their ego-motions. The experimental results show that our motion tracking system is able to cope with various types of ego-motions and the particle filter produces robust estimation. Second, a multiple particle filter approach for multiple motion tracking has been validated using various scenarios, and the experimental results show that the multiple filter approach tracks all motions successfully for the cases that a single filter approach fails. Lastly, the proposed tracking system has been integrated with a robot control loop, and its robustness and real-time capability have been examined. The experiments demonstrate that the motion tracking system is robust enough that the system is not disturbed by other moving objects once it starts to track an object.

The rest of the paper is organized as follows. Section II summarizes the related work on this topic. The detailed ego-motion compensation algorithm is given in Section III, and the design of the probabilistic filter is explained in Section IV. Section V describes how to fuse the estimation result in image space with laser rangefinder data, and Section VI reports the experimental results and analyzes the performance of the proposed algorithms. The current status and possible improvements are discussed in Section VII.

## II. Related Work

The computer vision community has proposed various methods to stabilize camera motions by tracking features [4], [5], [6] and computing optical flow [7], [8], [9]. These approaches focus on how to estimate the transformation ($homography$) between two image coordinate systems. However, the motions of moving objects are typically not considered, which leads to poor estimation.

Other approaches that extend these methods for motion tracking using a pan/tilt camera include those in [10], [11], [12]. However, in these cases the camera motion was limited to translation or rotation. When a camera is mounted on a mobile robot, the main motion of the camera is a forward/backward movement, which makes the problem different from that of a pan/tilt camera.

There is other research on tracking from a mobile platform with similar motions. [13] tracks a single object in forward-looking infrared (FLIR) imagery taken from an airborne, moving platform, and [14], [15] track cars in front using a camera mounted on a vehicle driven on a paved road.

Once motion has been identified, objects in the scene need to be tracked. Work focusing on robust multiple target tracking using probabilistic filters includes [2] which uses a particle filter to track people indoors (corridors) using a laser rangefinder, and [16] which also uses a particle filter to track multiple objects using a stationary camera. A Kalman filter was used in [17] to detect and track human activity with the combination of a static camera and a moving camera.

## III. Ego-motion Compensation

The ego-motion compensation is a coordinate conversion procedure. Assume that a sensory data acquisition process is as follows: (1) one set of data $D$ is acquired at time $t$ when a robot is located at $(x, y, \alpha)$, (2) the robot (and the sensors) moves to $(x + \Delta x, y + \Delta y, \alpha + \Delta\alpha)$ for $\Delta t$, and (3) another set of data $D'$ is acquired at time $t + \Delta t$. In this case, the data $D$ and $D'$ cannot be compared directly because they are captured in different coordinate systems. Therefore, the data $D$ should be compensated for the ego-motion $(\Delta x, \Delta y, \Delta\alpha)$, which means that the data $D$ should be transformed as if it were acquired when a robot was located at $(x + \Delta x, y + \Delta y, \alpha + \Delta\alpha)$. The goal of the ego-motion compensation step is to compute this transformation $T$. The transformation can be estimated directly or indirectly.

The indirect method is to estimate a robot pose each time using various sensors (*eg.* gyroscope, accelerometer, odometer and/or GPS), and compute the ego-motion $(\Delta x, \Delta y, \Delta\alpha)$ first. Once the ego-motion is computed, the transformation $T$ can be computed based on the geometric properties of sensors. The pose estimation technique has been well studied [18], [19], [20], and the indirect method may be effective when the transformation error is linear in the ego-motion estimation error. However, when a projection operation is involved in the transformation, as in our case, the indirect method is not appropriate since a tiny angular error in the motion estimation step would induce a huge position error after being projected into the data space. Therefore, we choose the direct method.

The direct method is to infer the transformation $T$ by corresponding salient features in the data set $D$ to those in the data set $D'$. The feature selection and matching techniques for various types of sensors has been studied by [4], [7], [21], [22], [23]. Since the transformation is estimated using the corresponding feature set directly, the quality of the transformation relies on the quality of selected features from the data sets. Unfortunately, in our case, the quality of the features is poor due to independently moving objects in image data. Therefore, a transformation model and outlier detection algorithm needs to be designed so that the estimated transformation is not sensitive to those object motions.

### A. Feature Selection and Tracking

We adopt the KLT (Kanade-Lucas-Tomasi) feature tracking algorithm introduced in [4], [7], [24], [25] to select and correspond features between two images. The KLT algorithm has become a standard technique for feature-based computer vision algorithms. For completeness, we describe the algorithms concisely here.

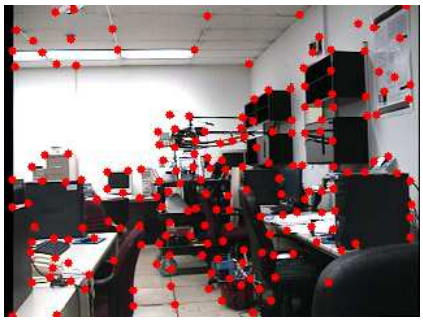
Given two consecutive images (the previous image $I^{t-1}$ and the current $I^t$), a set of features is selected from the image $I^{t-1}$, and a corresponding feature set is constructed by tracking the same features on the image $I^t$.

For feature selection, a small search window runs over the whole image $I^{t-1}$ to check if the window contains a *reliably trackable* feature. For each search window,

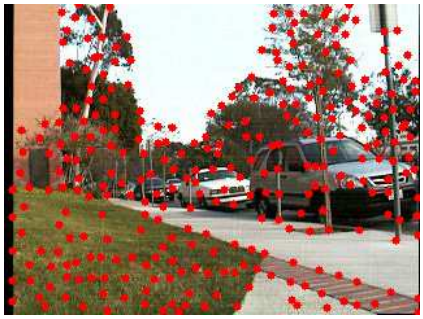
1) Compute the boundary information, $\begin{bmatrix} \frac{\partial I(x,y)}{\partial x} & \frac{\partial I(x,y)}{\partial y} \end{bmatrix}^T$
2) Compute the covariance matrix of the boundary pixels
3) Compute two eigenvalues $(\lambda_1, \lambda_2)$ of the covariance matrix
4) Select a search window such that $min(\lambda_1, \lambda_2) > \theta$

Search windows with two small eigenvalues contain no pattern, and those with one small eigenvalue and one big eigenvalue contain unidirectional patterns, which are not easy to track. Only search windows with two big eigenvalues are selected for tracking because they contain a perpendicular pattern (*eg.* corners) or divergent textures (*eg.* leaves) which are relatively unique enough to be tracked. The feature selection algorithm runs on the image $I^{t-1}$, and generates a set of features $F^{t-1}$. Figure 3 shows the features selected from indoor and outdoor images. In the indoor image, most of the selected features are the corners of objects, like desks, computers, and bookshelves. In the outdoor image, some corners of bricks and cars, leaves and grass that have complex textures were selected as features.

Once the feature set $F^{t-1}$ is selected, the features are tracked on the subsequent image $I^t$ and the set of tracked features $F^t$ is generated. For efficiency, the search range was limited to a small constant distance (assuming a bounded robot speed). Figure 4 shows the robustness of the tracking method. Figure 4 (a) shows the features selected from the image $I^t$, and Figure 4 (b) shows the same features tracked over 30 frames on the image $I^{t+30}$, which is an image captured 3 seconds later. The erroneous features on image boundaries are eliminated for subsequent processing.



(a) indoor features



(b) outdoor features

Fig. 3. Salient features (filled circles) selected for tracking: Primarily perpendicular patterns (*eg.* corners) or divergent textures (*eg.* leaves) are selected.

(a) features at time $t$

(b) tracked features at time $t + 30$

Fig. 4. Feature tracking: (a) shows salient features selected, and (b) shows the same features tracked over 30 frames (3 seconds).

### B. Transformation Estimation

Once the correspondence $F = < F^{t-1}, F^t >$ is known, the ego-motion of the camera can be estimated using a transformation model and an optimization method. We have studied three different models: affine model, bilinear model, and a pseudo-perspective model.

Affine Model :

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0\ f_x^{t-1} + a_1\ f_y^{t-1} + a_2 \\ a_3\ f_x^{t-1} + a_4\ f_y^{t-1} + a_5 \end{bmatrix}$$

Bilinear Model :

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0\ f_x^{t-1} + a_1\ f_y^{t-1} + a_2 + a_3\ f_x^{t-1}\ f_y^{t-1} \\ a_4\ f_x^{t-1} + a_5\ f_y^{t-1} + a_6 + a_7\ f_x^{t-1}\ f_y^{t-1} \end{bmatrix}$$

Pseudo-perspective Model :

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0\ f_x^{t-1} + a_1\ f_y^{t-1} + a_2 \\ \qquad + a_3\ {f_x^{t-1}}^2 + a_4\ f_x^{t-1}\ f_y^{t-1} \\ a_5\ f_x^{t-1} + a_6\ f_y^{t-1} + a_7 \\ \qquad + a_4\ f_x^{t-1}\ f_y^{t-1} + a_3\ {f_y^{t-1}}^2 \end{bmatrix}$$

When the interval between consecutive images is very small, most ego-motions of the camera can be estimated using an affine model, which can cover translation, rotation, shearing, and scaling motions. However, when the interval is long, the camera motion in the interval cannot be captured by a simple linear model. For example, when the robot moves forward, the features in the image center move slower that those near the image boundary, which is a projection operation, not a simple scaling. Therefore, a nonlinear transformation model is required for those cases. On the other hand, an over-fitting problem may be caused when a model is highly nonlinear, especially when some of the selected features are associated with moving objects (*outliers*). There is clearly a trade-off between a simple, linear model and a highly nonlinear model, and it needs more empirical research for the best selection. We used a bilinear model for the experiments reported in this paper.

When the transformation from the image $I^{t-1}$ to the image $I^t$ is defined as $T_{t-1}^t$, the cost function for least square optimization is defined as:

$$J = \frac{1}{2} \sum_{i=1}^{N} \left\{ f_i^t - T_{t-1}^t \left( f_i^{t-1} \right) \right\}^2 \tag{1}$$

where $N$ is the number of features. The model parameters for ego-motion compensation are estimated by minimizing the cost. However, as mentioned before, some of the features are associated with moving objects, which lead to the inference of an inaccurate transformation. Those features (outliers) should be eliminated from the feature set before the final transformation is computed. The model parameter estimation is thus performed using the following two-step procedure:

1) compute the initial estimate $T_0$ using the full feature set $F$.
2) partition the feature set $F$ into two subsets $F_{in}$ and $F_{out}$ as:

$$\begin{cases} f_i \in F_{in} & \text{if } |f_i^t - T_{0\,t-1}^t(f_i^{t-1})| < \epsilon \\ f_i \in F_{out} & \text{otherwise} \end{cases} \tag{2}$$

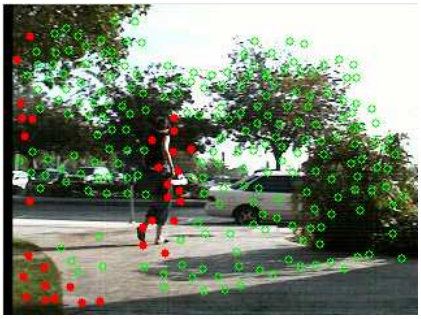


Fig. 5. Outlier feature detection: Outliers are marked with filled circles, and inliers are marked with empty circles.

3) re-compute the final estimate $T$ using the subset $F_{in}$ only.

Figure 5 shows the partitioned feature sets: $F_{in}$ is marked with empty circles, and $F_{out}$ is marked with filled circles. Note that all features associated with the pedestrian are detected as outliers. It is assumed for outlier detection that the portion of moving objects in the images is relatively smaller compared to the background; the features which do not agree with the main motion are considered as outliers. This assumption will break when the moving objects are very close to the camera. However, most of the time, these objects pass by the camera in a short period (leading to transient errors), and a high-level probabilistic filter is able to deal with the errors without total failure.

### C. Frame Differencing

The image $I^{t-1}$ is converted using the transformation model before being compared to the image $I^t$ in order to eliminate the effect of the camera ego-motion. For each pixel $(x, y)$:

$$I_c(x, y) = I^{t-1}\left( {T_{t-1}^t}^{-1}(x, y) \right) \tag{3}$$

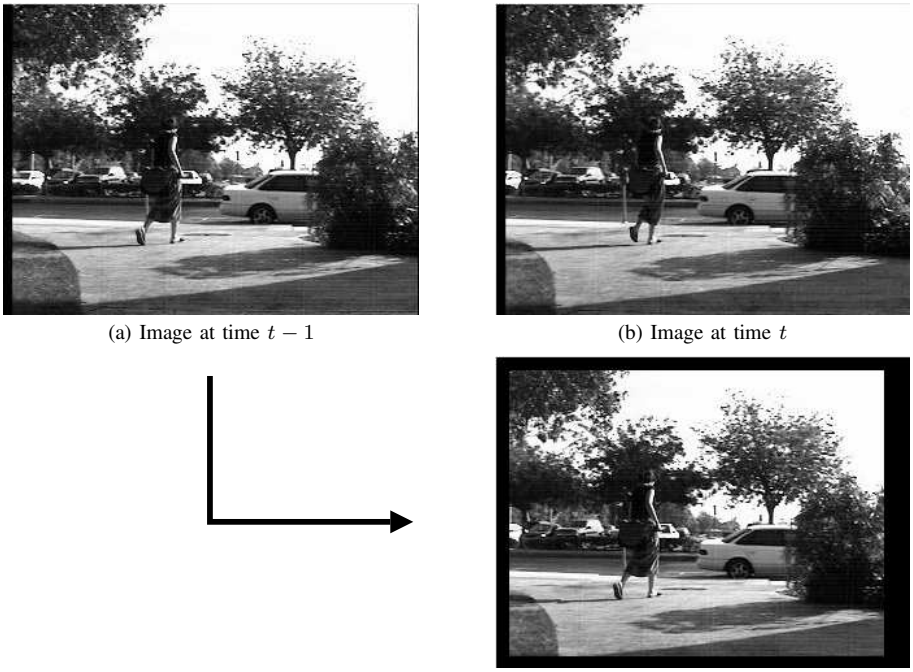
Figure 6 (c) shows the compensated image of Figure 6 (a); the translational and forward motions of the camera were clearly eliminated. The valid region $\Re$ of the transformed image is smaller than that of the original image because some pixel values on the border are not available in the original image $I^{t-1}$. The invalid region in Figure 6 (c) is filled black. The difference image between two consecutive images is computed using the compensated image:

$$I_d(x, y) = \begin{cases} |\left( I_c(x, y) - I^t(x, y) \right)| & \text{if } (x, y) \in \Re \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Figure 7 compares the results of two cases: frame differencing without ego-motion compensation (Figure 7 (a)) and with ego-motion compensation (Figure 7 (b)). The results show that the ego-motion of the camera is decomposed and eliminated from image sequences. The full description of the frame differencing process is given in Algorithm 1.

## IV. Motion Detection in 2D Image Space

The *Frame Differencing* step in Figure 2 generates the sequence of difference images, $I_d^0, I_d^1, \cdots, I_d^t$, whose pixel

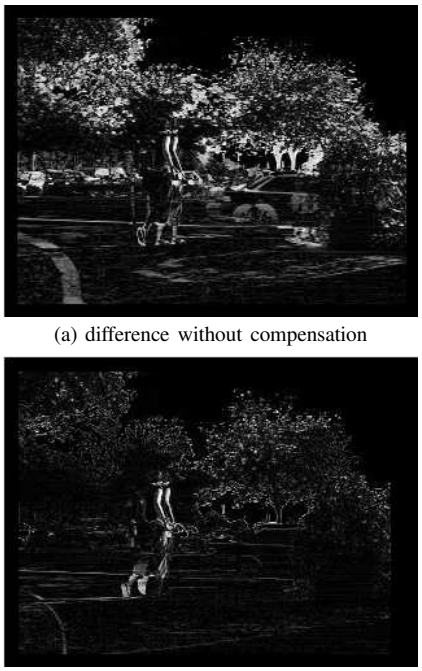


(a) Image at time $t-1$

(b) Image at time $t$

(c) Compensated image of (a)

Fig. 6. Image Transformation: (c) is the transformed image of (a) into (b) coordinates. The valid region of the compensated image is smaller than that of the original image due to the absence of data on the border.



(a) difference without compensation

(b) difference with compensation

Fig. 7. Results of frame differencing: (b) shows that the ego-motion of a camera was decomposed and eliminated from image sequences.

values represent the amount of motion occurred in the position. However, as described earlier, the difference images contain two different types of errors. There are transient errors caused by imperfect ego-motion compensation, and this type of error should be filtered out using their temporal properties. There are also persistent errors caused during data acquisition. Since the camera positions are different when two consecutive images are captured, it is inevitable that some information is newly introduced to the current image $I^t$ or some information of the previous image $I^{t-1}$ is occluded in the image $I^t$.

To deal with those errors, a probabilistic approach is adopted. The normalized pixel values in the difference images can be interpreted as the probability of the existence of moving objects in that position, and the position and size of the moving objects are estimated over time. This estimation process can be modeled using a Bayesian formulation. Let $\mathbf{x}^t$ represent the state (*eg.* the position and velocity) of a moving object.

$$\mathbf{x} = [\ x\ \ y\ \ \dot{x}\ \ \dot{y}\ ]^T \tag{5}$$

The posterior probability distribution $P_m(\mathbf{x}^t)$ of the state is derived as follows.

$$\begin{aligned} P_m(\mathbf{x}^t) &= P(\mathbf{x}^t|I_d^0, I_d^1, \cdots, I_d^t) \\ &= \eta^t\ P(I_d^t|\mathbf{x}^t, I_d^0 \cdots, I_d^{t-1})\ P(\mathbf{x}^t|I_d^0 \cdots, I_d^{t-1}) \\ &= \eta^t\ P(I_d^t|\mathbf{x}^t)\ P(\mathbf{x}^t|I_d^0 \cdots, I_d^{t-1}) \\ &= \eta^t\ P(I_d^t|\mathbf{x}^t)\ \int P(\mathbf{x}^t|I_d^0 \cdots, I_d^{t-1}, \mathbf{x}^{t-1}) \\ &\qquad \times P(\mathbf{x}^{t-1}|I_d^0 \cdots, I_d^{t-2})\ d\mathbf{x}^{t-1} \\ &= \eta^t\ P(I_d^t|\mathbf{x}^t)\ \int P(\mathbf{x}^t|\mathbf{x}^{t-1}) \\ &\qquad \times P(\mathbf{x}^{t-1}|I_d^0 \cdots, I_d^{t-2})\ d\mathbf{x}^{t-1} \\ &= \eta^t\ P(I_d^t|\mathbf{x}^t)\ \int P(\mathbf{x}^t|\mathbf{x}^{t-1}) P_m(\mathbf{x}^{t-1})\ d\mathbf{x}^{t-1} \end{aligned} \tag{6}$$

Now the posterior probability distribution can be updated recursively by applying a perception model $P(I_d^t|\mathbf{x}^t)$ and a motion model $P(\mathbf{x}^t|\mathbf{x}^{t-1})$ over time.

**Algorithm 1**: Frame Differencing with Ego-motion Compensation

**Input**: two consecutive images $I^{t-1}$ and $I^t$
**Output**: the difference image $I_d^t$ and the ego-motion transformation $T_{t-1}^t$

**begin**
  select a salient feature set $F^{t-1}$ from $I^{t-1}$ ;
  track the features on $I^t$, and generate the corresponding feature set $F^t$ ;
  $F \leftarrow < F^{t-1},\ F^t >$ ;
  **if** $|F| >$ *the number of parameters in a transformation model* **then**
    */* compute the initial estimate using the full feature set. */*
    construct an input matrix $X$ using $F^{t-1}$ ;
    construct a target vector $t$ using $F^t$ ;
    $T_0 \leftarrow \left(X^T X\right)^{-1} X^T t$
    */* remove outliers from the feature set. */*
    $F_{in} \leftarrow \emptyset$ ;
    $F_{out} \leftarrow \emptyset$ ;
    **foreach** $< f^{t-1},\ f^t >$ *in the set* $F$ **do**
      **if** $|f^t - T_0(f^{t-1})| < \epsilon$ **then**
        insert $< f^{t-1},\ f^t >$ into $F_{in}$;
      **else**
        insert $< f^{t-1},\ f^t >$ into $F_{out}$;
      **end**
    **end**
    */* compute the final estimate using the inliers only. */*
    construct an input matrix $X$ using only $F_{in}^{t-1}$ ;
    construct a target vector $t$ using only $F_{in}^t$ ;
    $T_{t-1}^t \leftarrow \left(X^T X\right)^{-1} X^T t$
    */* compensate ego-motion. */*
    initialize $I_c$ with all 0 ;
    **forall** $(x, y)$ *in the image* $I_c$ **do**
      $I_c(x,y) \leftarrow I^{t-1}\left({T_{t-1}^t}^{-1}(x,y)\right)$ ;
    **end**
    */* frame differencing with ego-motion compensation. */*
    $I_d^t \leftarrow |I^t - I_c|$ ;
  **else**
    */* no compensation is feasible without sufficient features. */*
    $T_{t-1}^t \leftarrow$ the identical transformation ;
    $I_d^t \leftarrow |I^t - I^{t-1}|$ ;
  **end**
  **return** $I_d^t$ *and* $T_{t-1}^t$
**end**

### A. Bayesian Filter Design

The derived equation 6 shows how the sequence of difference images and the motion model of a moving object are integrated into the state estimation process. However, there are still three questions to answer: (1) how to define a perception model, and (2) how to define a motion model, and finally (3) how to represent the posterior probability distribution.

The perception model $P(I_d^t|\mathbf{x}^t)$ captures the idea that if there is a motion at position $\mathbf{x}^t$, then the difference values of the pixel in that position and its neighbors should be big. For example, let us assume a small moving object that occupies a single pixel $\mathbf{p}^{t-1}$ on a camera image. When the object moves to its neighbor pixel $\mathbf{p}^t$, then the difference values of both pixels $\mathbf{p}^{t-1}$ and $\mathbf{p}^t$ would be big. This *neighborhood* can be modeled using a multi-variate Gaussian, and the perception model can be defined as

$$P(I_d^t \,|\, \mathbf{x}^t) = \int_{\mathbf{0}}^{|I_d^t|} I_d^t(\mathbf{x}) \times \frac{1}{\sqrt{(2\pi)^d |\Sigma_s|}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{x^t})^T \Sigma_s^{-1} (\mathbf{x}-\mathbf{x^t})}\, d\mathbf{x} \tag{7}$$

when $d$ is the dimension of the state $\mathbf{x}$, and the covariance matrix $\Sigma_s$ controls the range of effective neighborhood.

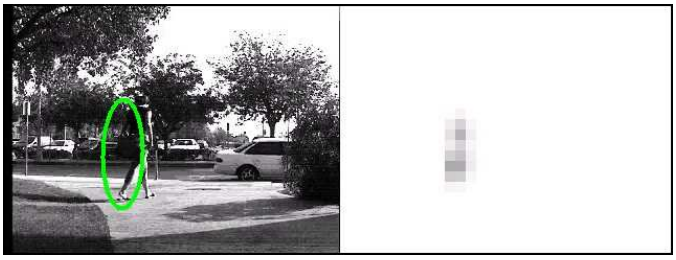


Fig. 8. Bayesian filter tracking with piecewise constant representation: The left window shows the input image and the detected moving object (ellipsoid), and the right window shows the posterior probability distribution of the moving object using a 10x10 pixel grid.

The motion model $P(\mathbf{x}^t|\mathbf{x}^{t-1})$ captures the best guess about the motion of a moving object. Since no prior knowledge of an object motion is assumed, a constant velocity model is a natural choice. The uncertainty of an object motion is modeled by the covariance matrix $\Sigma_m$ of a multi-variate Gaussian.

$$\mu = \begin{bmatrix} x^{t-1} + \Delta t \times \dot{x}^{t-1} \\ y^{t-1} + \Delta t \times \dot{y}^{t-1} \\ \dot{x}^{t-1} \\ \dot{y}^{t-1} \end{bmatrix} \tag{8}$$

$$P(\mathbf{x}^t \,|\, \mathbf{x}^{t-1}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_m|}} e^{-\frac{1}{2}(\mathbf{x^t}-\mu)^T \Sigma_m^{-1} (\mathbf{x^t}-\mu)} \tag{9}$$

The choice of a representation for the posterior probability distribution is important for real-time system design because the update equation (Equation 6) contains an integral operation and the required computation is intensive. Even when the size of a camera image is small, the state space is sizeable because the state is four-dimensional. The most compact representation is to use a single Gaussian, like the Kalman filter [26], [27], but this approach is not appropriate because (1) the initial state of a moving object is not given in priori, and (2) image segmentation is avoided for real-time response, which makes it hard to construct a measurement matrix.

A better approach is to use a piecewise constant representation. By decomposing the state space into an equally spaced grid, the amount of computation can be reduced drastically. For example, Figure 8 shows the position estimation result using a 10x10 pixel grid. However, the computation was still not efficient enough. In order to achieve a real-time response, only the position of a moving object was estimated ($\mathbf{x} = [x\ y]^T$), and a constant position model, instead of a constant velocity model, was utilized as the consequence of the simpler state definition. In addition, the approximation quality of the posterior probability distribution was sacrificed by using the sparse representation.

The most popular representation that addresses these concerns is a sample-based representation [28], [29]. The amount of computation is reduced by using a small set of weighted samples, but the approximation quality is well preserved by concentrating samples on the area whose probability is high. Also, the number of samples can change dynamically depending on the shape of the posterior probability distribution and available computer power. In this paper, we adopt the sample-based representation.

### B. Particle Filter Design

The *Particle filter* [28], [29] is a simple but effective algorithm to estimate the posterior probability distribution recursively, which is appropriate for real-time applications. In addition, its ability to perform multi-modal tracking is attractive for unsegmented object detection and tracking from camera images. An efficient variant, called the *Adaptive Particle Filter*, was introduced in [30]. This changes the number of particles dynamically for a more efficient implementation. We implemented the *Adaptive Particle Filter* to estimate the posterior probability distribution in Equation 6. As described in Section IV-A, particle filters also require two models for the estimation process: a perception model and a motion model.

The perception model is used to evaluate a particle and compute its weight (or importance). Equation 7 provides a generic form of the perception model. However, the perception model is simplified for efficiency. A step function is used instead of a multi-variate Gaussian, and the evaluation range is also limited to $m \times m$ fixed area. The $m \times m$ mask should be big enough (usually $5 \times 5$) so that salt-and-pepper noise is eliminated. The weight $\omega_i^t$ of the $i_{th}$ particle ($s_i^t = [\, x_i^t \;\; y_i^t \;\; \dot{x}_i^t \;\; \dot{y}_i^t \,]^T$) is computed by

$$\omega_i^t = \frac{1}{m^2} \sum_{j=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} I_d \left( x_i^t - j, y_i^t - k \right) \qquad (10)$$

As shown in Equation 10, only the position information is used to evaluate particles.

The motion model is used to propagate a newly drawn particle according to the estimated motion of a moving object. The motion model in and Equation 9 describes how to compute the probability of the new state $\mathbf{x^t}$ when the previous state $\mathbf{x^{t-1}}$ is given. However, for particle filter update, the motion model should describe how to draw a new particle $s_i^t$ when the previous particle $s_i^{t-1}$ and its weight $\omega_i^t$ are given. Therefore, the motion model is defined as

$$s_i^t = \begin{bmatrix} x_i^{t-1} + \Delta t \times \dot{x}_i^{t-1} & + Normal(\frac{\gamma_p}{\omega_i^t}) \\ y_i^{t-1} + \Delta t \times \dot{y}_i^{t-1} & + Normal(\frac{\gamma_p}{\omega_i^t}) \\ \dot{x}_i^{t-1} & + Normal(\frac{\gamma_v}{\omega_i^t}) \\ \dot{y}_i^{t-1} & + Normal(\frac{\gamma_v}{\omega_i^t}) \end{bmatrix} \qquad (11)$$

where $\Delta t$ is a time interval, and $\gamma_p$ and $\gamma_v$ are noise parameters for position and velocity components respectively. The function $Normal(\sigma)$ generate a Gaussian random variate with zero mean and the standard deviation $\sigma$. As shown in Equation 11, the parameterized noise is added to the constant-velocity model in order to overcome an intrinsic limitation of the particle filter, which is that all particles move in a convergence direction. However, a dynamic mixture of divergence and convergence is required to detect newly introduced moving objects. [29] introduced a mixture model to solve this problem, but in the image space the probability $P(\mathbf{x}^t | I_d^t)$ is uniform and the dual MCL becomes random. Therefore, we used a simpler, but effective method by adding inverse-proportional noise.

As an implementation issue, a multi-dimensional kd-tree is constructed during the particle filter update. This serves two purposes: (1) to compute the proper number of particles [30], and (2) to cluster particles efficiently as described in Section IV-C. In order to determine the proper number of particles, a kd-tree with uniform-size nodes is built. When the size of the tree is $k$, the error bound is $\epsilon$, and the confidence quantile is $z_{1-\delta}$, the proper number is computed as follows [30].

$$\begin{aligned} n &= \frac{1}{2\epsilon} \chi^2_{k-1,1-\delta} \\ &\doteq \frac{k-1}{2\epsilon} \left\{ 1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}\, z_{1-\delta} \right\}^3 \end{aligned} \qquad (12)$$

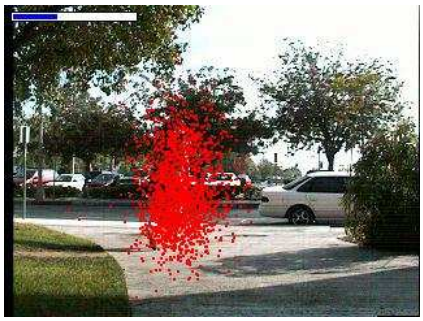


Fig. 9. Particle filter tracking: The position of particles are represented by small dots, and the horizontal bar on the top-left corner shows the number of particles being used.

Figure 9 shows the output of the particle filter. The dots represent the position of particles, and the horizontal bar on the top-left corner of the image shows the number of particles being used. The final algorithm of the particle filter is described in Algorithm 2.

### C. Particle Clustering

The particle filter generates a set of weighted particles that estimate the posterior probability distribution of a moving object, but the particles are not easy to process in the following step. More intuitive and meaningful data can be extracted by clustering the particles. A density-based algorithm using a kd-tree is introduced for efficient particle clustering. The main idea is to convert a set of weighted particles into a lower-resolution, uniform-sized grid. The grids can be represented using a kd-tree efficiently, and all clustering operations are performed using the grids instead of particles. Therefore, the required computation is reduced drastically. However, since each grid maintains enough information about the particles in the grid, the statistics of each cluster can be calculated without any accuracy loss. The algorithm consists of the following four steps:

*1) Tree Construction:* Given a set of weighted particles, a kd-tree representation is constructed. The state space is partitioned into uniform-sized grid cells, and only non-empty cells are maintained using a kd-tree. Since the *Adaptive Particle Filter* requires the kd-tree for computing the proper number of particles (as described in Section IV-B), this step can be combined with the particle filter update. The information of each particle is not necessary anymore for the subsequent steps, but a few extra statistics of each terminal node in the

**Algorithm 2:** Adaptive Particle Filter for Motion Tracking

**Input**: the previous particle set $S^{t-1}$, the difference image $I_d^t$, and the ego-motion transformation $T_{t-1}^t$
**Output**: a new particle set $S^t$ and its kd-tree representation $K^t$

$S^0 \leftarrow$ a set of uniformly weighted, random particles with the size $n_{max}$ ;
**begin**
  */* transform all particles to compensate an ego-motion */*
  **foreach** $< s^{t-1}, \omega^{t-1} >$ *in* $S^{t-1}$ **do**
    $s^{t-1} \leftarrow T_{t-1}^t(s^{t-1})$ ;
  **end**
  */* perform the standard particle filter update */*
  $S^t \leftarrow \emptyset$ ;
  $K^t \leftarrow \emptyset$ ;
  $W \leftarrow 0$ ;
  **repeat**
    */* draw a particle from $S^{t-1}$ according to its weight values */*
    $C_1 \leftarrow \omega_1^{t-1}$ ;
    **for** *i=2* **to** $|S^{t-1}|$ **do** $C_i \leftarrow C_{i-1} + \omega_i^{t-1}$ ;
    generate a random number $r$ in the range $[0, 1)$ ;
    select the $i_{th}$ particle $s_i'$ from $S^{t-1}$ such that $C_i \le r < C_{i+1}$ ;
    */* propagate the particle using the motion & sensor models */*
    $\omega' \leftarrow \frac{1}{m^2} \sum_{j=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} I_d^t\left(x_i' - j, y_i' - k\right)$ ;
    $$s^t \leftarrow \begin{bmatrix} x_i' + \Delta t \times \dot{x}_i' & + Normal(\frac{\gamma_p}{\omega'}) \\ y_i' + \Delta t \times \dot{y}_i' & + Normal(\frac{\gamma_p}{\omega'}) \\ \dot{x}_i' & + Normal(\frac{\gamma_v}{\omega'}) \\ \dot{y}_i' & + Normal(\frac{\gamma_v}{\omega'}) \end{bmatrix} ;$$
    $\omega^t \leftarrow \frac{1}{m^2} \sum_{j=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} I_d^t\left(x^t - j, y^t - k\right)$ ;
    */* add the new particle */*
    add $< s^t, \omega^t >$ to $S^t$ ;
    add $< s^t, \omega^t >$ to $K^t$ ;
    $W \leftarrow W + \omega^t$ ;
  **until** $|S^t| < n_{min}$ *or* $|S^t| < \frac{|K^t|-1}{2\epsilon}\left\{1 - \frac{2}{9(|K^t|-1)} + \sqrt{\frac{2}{9(|K^t|-1)}} z_{1-\delta}\right\}^3$ ;
  */* normalize the weights of all particles */*
  **if** $W > 0$ **then**
    **foreach** $< s^t, \omega^t >$ *in* $S^t$ **do** $\omega^t \leftarrow \omega^t / W$ ;
  **else**
    $S^t \leftarrow$ a set of uniformly weighted, random particles with the size $n_{max}$ ;
  **end**
  **return** $S^t$ *and* $K^t$
**end**

tree need to be computed. For each terminal node $k$, the weight $w_k$, the mean $\mu_k$, and the covariance matrix $\Sigma_k$ of the node are calculated using the subset of particles that are associated with the node.

$$\begin{aligned} w_k &= \textstyle\sum_i w_i \\ \mu_k &= \textstyle\sum_i w_i s_i \;/\; \sum_i w_i \\ \Sigma_k &= \textstyle\sum_i w_i (s_i - \mu)(s_i - \mu)^T \;/\; \sum_i w_i \end{aligned} \tag{13}$$

*2) Candidate Selection:* Instead of re-constructing a posterior probability distribution and thresholding the *pdf*, we select candidate grid cells whose particle density is bigger than a threshold $\theta$. Since each particle has different weight, the density should take the weight into account. Theoretically this means $w_k/volume(k)$ should be used as the determinant. However, the number of particles ($n_k/volume(k)$) can be used alternatively for simplicity assuming all particles have uniform weights.

Fig. 10. Particle clustering: Two ellipsoids represent the means and covariance of two particle clusters.

*3) Grouping:* Once the candidate nodes are selected, clustering can be done by simply grouping the nodes by checking connectivity among nodes. There are various known algorithms for this task. The connectivity can be defined using the distance between the mean vectors of two nodes, or can be determined by checking if a node is a neighbor of another.

*4) Statistics Computation:* For each cluster, the statistics of the particles in the cluster can be calculated by summing the statistics of the nodes in the cluster incrementally.

$$\begin{aligned} \mu' &= \tfrac{w\,\mu + w_k\,\mu_k}{w + w_k} \\ \Sigma &= \tfrac{w\left\{\Sigma + (\mu' - \mu)(\mu' - \mu)^T\right\} + w_k\left\{\Sigma_k + (\mu' - \mu_k)(\mu' - \mu_k)^T\right\}}{w + w_k} \\ w &= w + w_k \\ \mu &= \mu' \end{aligned} \tag{14}$$

Figure 10 shows the output of the particle clustering algorithm. The dots represent the position of particles, and the ellipsoid represents the mean and covariance of each cluster. The full description of the clustering algorithm is in Table 3.

### D. Multiple Particle Filters for Multiple Motion Tracking

The particle filter has many advantages as described in [31], [29]; one of the advantages is multi-modality. This property is attractive for multi-target tracking because it raises the possibility that a single set of particles can track multiple objects in an image sequence. However, that is true only under two conditions:

1) *The perception model should be "bad" enough* so that particles converge slowly, and eventually stay on multiple objects. For example, imagine the case in which there are two moving objects in the image sequence. It is desired that a *single* set of particles is split into two groups, and each group converges to and track each objects continuously. Apparently a set of particles would behave as desired if two objects show exactly the same amount of motions (on average over time) in the image sequence. However, this assumption is not realistic. In most cases, one would show a "bigger" motion than the other due to different size and shape of an object, different distance to a camera, etc. The behavior of particles is quite different without the assumption. Since the amount of motion is different, the perception model $P(I_d^t|\mathbf{x}^t)$ in Equation 6 generates a different value for

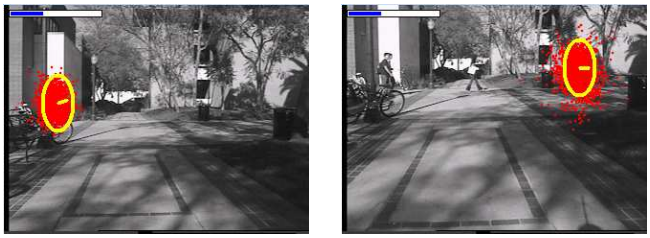
**Algorithm 3**: Particle Clustering for Post-processing

**Input**: the kd-tree representation $K$ of a particle set
**Output**: $m$ Gaussians $(< \mu_1, \Sigma_1 >, \cdots, < \mu_m, \Sigma_m >)$

**begin**
  */* select a set of candidate nodes $C$ based on their density */*
  $C \leftarrow \emptyset$ ;
  **forall** *a terminal node $k$ in $K$* **do**
    the particles in the node $k$ are $< s_1, w_1 >, < s_2, w_2 >, \cdots, < s_n, w_n >$ ;
    **if** $\sum_i w_i \,/\, \mathtt{volume}(k) \geq \theta$ */* thresholding by weighted density */*
    **then**
      */* these statistics can be computed when the tree is built */*
      $w_k \leftarrow \sum_i w_i$ ;
      $\mu_k \leftarrow \sum_i w_i s_i \;/\; \sum_i w_i$ ;
      $\Sigma_k \leftarrow \sum_i w_i (s_i - \mu)(s_i - \mu)^T \;/\; \sum_i w_i$ ;
      add $< w_k, \mu_k, \Sigma_k >$ to $C$ ;
    **end**
  **end**
  */* group the candidate nodes using connectivity */*
  **for** $i = 1$ *to* $|C|$ **do**
    **if** $c_i \notin$ *any group* **then** create a new group $G_i$ ;
    **for** $j = i + 1$ *to* $|C|$ **do**
      **if** $c_j \notin$ *any group* **then**
        **if** $|\mu_i - \mu_j| \leq \rho$ **then** add $c_j$ to the group $G_i$
      **else if** $G_i \neq G_j$ *and* $|\mu_i - \mu_j| \leq \rho$ **then**
        combine the group $G_i$ and the group $G_j$ ;
      **end**
    **end**
  **end**
  */* compute the statistics of each group */*
  **foreach** *a group $G_i$* **do**
    $w_i \leftarrow 0$ ;
    $\mu_i \leftarrow 0$ ;
    $\Sigma_i \leftarrow 0$ ;
    **forall** *a candidate* $< w_k, \mu_k, \Sigma_k >$ *in* $G_i$ **do**
      $\mu' \leftarrow \frac{w_i \mu_i + w_k \mu_k}{w_i + w_k}$ ;
      $\Sigma_i \leftarrow \frac{w_i \{\Sigma_i + (\mu' - \mu_i)(\mu' - \mu_i)^T\} + w_k \{\Sigma_k + (\mu' - \mu_k)(\mu' - \mu_k)^T\}}{w_i + w_k}$ ;
      $w_i \leftarrow w_i + w_k$ ;
      $\mu_i \leftarrow \mu'$
    **end**
  **end**
**end**
**return** $< \mu_1, \Sigma_1 >, \cdots, < \mu_m, \Sigma_m >$

each object. As a result, particles on the smaller motion would shift to the bigger motion after some iterations even when the size of the two particle groups was the same in the beginning. This behavior is expected because Equation 6 is designed to estimate the position of a single object $\mathbf{x}$. This limitation can be overcome in two ad-hoc ways: (1) By making the perception model less sensitive so that particles converge very slowly, (2) By increasing the number of particles. The first technique is feasible when the convergence speed is not important [32], [29]. However, convergence speed is a critical factor for motion tracking. When a new object is introduced, a filter should be able to detect it and start to track it in a reasonable time. The second technique is not desirable since the required computation increases drastically.

2) *All objects should be introduced in the beginning of estimation process.* As explained in Section IV-A, particles shows a convergence tendency, and consequently converged particles do not diverge unless the tracked object disappears. For example, Figure 11 shows a problematic scenario. The particle set converges when the person enters into the field of view of the camera as in Figure 11 (a), and it concentrates on the person continuously even when two more people enter later as in Figure 11 (b). This problem is not trivial to solve using a limited number of particles.



(a) single person at $t = 23$ (b) three people at $t = 109$

Fig. 11. Problem of a single particle filter: (a) shows that the particle filter converges on the person, and (b) shows that the filter stays on the person continuously even when other people are introduced in the image later.

Therefore, we introduce a tracking system using multiple particle filters. The main idea is to maintain an extra particle filter for a newly introduced or detected object. Since the number of objects is not known in priori, particle filters should be created and destroyed dynamically. Whenever the extra particle filter converges on a newly detected object, a new particle filter is created (as long as the number of particle filters is smaller than the maximum limit $N_{max}$). Similarly, whenever a particle filter diverges due to the disappearance of a tracked object, it is destroyed. In order to prevent two particle filters from converging on the same object, whenever a particle filter is updated, the difference image is modified for subsequent processing such that difference values covered by the filter are cleared. The detailed algorithm is described in Algorithm 4.

## V. Position Estimation in 3D Space

A monocular image provides rich information for ego-motion compensation and motion tracking in 2D image space. However, a single camera has limits on retrieving depth information, and an additional sensor is required to construct 3D models of moving objects. Our robots are equipped with a laser rangefinder, which provides depth information within a singe plane. Given the optical properties of a camera and the transformation between the camera and the laser rangefinder, distance information from the laser rangefinder can be projected onto the image coordinates (Figure 12).

Given the heading $\alpha$ and the range $r$ of a scan, the projected position $(x, y)$ in the image coordinate system is computed as follows:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{w}{2} \times \left(1 - \frac{\tan(\alpha)}{\tan(f_h)}\right) \\ \frac{h}{2} \times \left(1 + \left(d - \frac{d}{r} \times (r - l)\right) \times \frac{1}{l \times \tan(f_v)}\right) \end{bmatrix} \quad (15)$$

where the focal length of the camera is $l$, the horizontal and vertical field of view of the camera are $f_h$ and $f_v$, the

**Algorithm 4**: Multiple Particle Filters for Multiple Motion Tracking

**Input**: the previous particle filter set $P^{t-1}$, the difference image $I_d^t$, and the ego-motion transformation $T_{t-1}^t$
**Output**: a new particle filter set $P^t$

create a particle filter $p_0$ ;
$P^0 \leftarrow \{p_0\}$ ;
**begin**
  */* update all particle filters in the set */*
  $P^t \leftarrow \emptyset$ ;
  $I_d \leftarrow I_d^t$ ;
  **foreach** *a particle filter* $p_i$ *in* $P^{t-1}$ **do**
    update the particle filter $p_i$ using the difference image $I_d$ ;
    retrieve the cluster information $C_i$ from the updated particle filter $p_i$ ;
    **if** $|C_i| > 0$ */* check the convergence */*
    **then**
      $P^t \leftarrow P^t \cup \{p_i\}$ ;
      $P^{t-1} \leftarrow P^{t-1} - \{p_i\}$ ;
      **forall** *a cluster* $< \mu_j, \Sigma_j >$ *in* $Ci$ **do**
        $I_d \leftarrow I_d - \texttt{region}(\mu_j, \Sigma_j)$ ;
      **end**
    **end**
  **end**
  */* add an extra particle filter if allowed */*
  **if** $|P^{t-1}| > 0$ **then**
    select a particle filter $p'$ from $P^{t-1}$ ;
    reset the particle filter $p'$ ;
    $P^t \leftarrow P^t \cup \{p_i\}$ ;
    $P^{t-1} \leftarrow P^{t-1} - \{p_i\}$ ;
  **else**
    **if** $|P^t| < N_{max}$ **then**
      create a new particle filter $p'$ ;
      $P^t \leftarrow P^t \cup \{p_i\}$ ;
    **end**
  **end**
  */* destroy unused particle filters */*
  **forall** *a particle filter* $p_i$ *in* $P^{t-1}$ **do**
    destroy the particle filter $p_i$ ;
  **end**
**end**
**return** $P^t$

height from the laser rangefinder to the camera is $d$, and the image size is $w \times h$. This projection model assumes a very simple camera model (a pin-hole camera) for fast computation. As a result of the projection, the image pixels at the same height as the laser rangefinder will have depth information as shown in Figure 13. For ground robots, this partial 3D information can be enough for safe navigation assuming all moving obstacles are on the the same plane as the robot. In terms of moving object tracking, if the region of a moving objects in image space and those pixels are overlapped, then the distance between a robot and the moving object can be estimated.

This naive integration of the 2D motion estimates and range scans from a laser rangefinder is a reasonable practical solution. However, using two separate sensors requires another estimation problem potentially, which is the fusion of multiple asynchronous inputs. A preferred route (not investigated here) would be to use stereo vision for depth information retrieval. If computational power allows one can exploit the facts that stereo (1) provides full depth information of an image space, and (2) a single input source provides synchronous data and better fusion result can be expected.

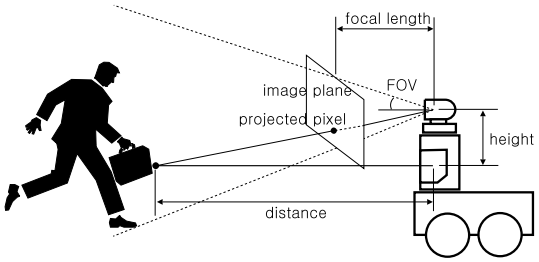



Fig. 12. Projection of laser scans onto the image coordinates: The range scans from a laser rangefinder can be projected onto the image coordinate system based on the optical properties of a camera and the transformation between the camera and the laser rangefinder.

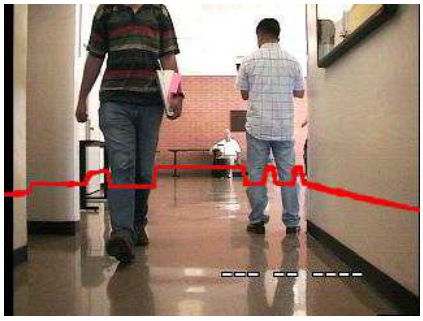


Fig. 13. Projected laser scans: The image pixels at the same height as the laser rangefinder have depth information.

## VI. Experimental Results and Discussion

The proposed motion tracking system was tested in various scenarios. First, the robustness of ego-motion compensation and motion tracking algorithms was tested using three different robot platforms. The performance is analyzed in Section VI-A. Second, the multiple-motion tracking system described in Section IV-D was tested using various scenarios. The results are presented in Section VI-B. Finally, the tracking system was integrated with an actual robot control system. The result is discussed in Section VI-C.

### A. Tracking a Moving Object from Various Platforms

The ego-motion of a mobile robot is diverse according to its actuator design and the way the camera is mounted on the platform. For example, a down-facing camera mounted on an UAV (Unmaned Aerial Vehicle) would show a different ego-motion from a forward-facing camera mounted on a walking robot. In addition, the complexity of the ego-motion increases through the interaction with rough terrain. The tracking performance is also affected by the distribution of occlusive obstacles in an environment. Therefore, the ego-motion compensation and the motion tracking algorithms should be tested on various environments with a wide variety of mobile platforms.

*1) Experimental Setup:* The tracking algorithms were implemented and tested in various outdoor environments using three different robot platforms: robotic helicopter, Segway RMP, and Pioneer2 AT. Each platform has unique characteristics in terms of its ego-motion.

The *Robotic Helicopter* [33] in Figure 14 (a) is an autonomous flying vehicle carrying a monocular camera facing


(a) Robotic Helicopter


(b) Segway RMP


(c) Pioneer2 AT

Fig. 14. Robot platforms for experiments: Each platform has unique characteristics in terms of its ego-motion.

downward. Once it takes off and hovers, planar movements are the main motion, and moving objects on the ground stay at a roughly constant distance from the camera most of the time; however, pitch and roll motions still generate complicate video sequences. Also, high-frequency vibration of the engine adds motion-blur to camera images.

The *Segway RMP* in Figure 14 (b) is a two-wheeled, dynamically stable robot with self-balancing capability. It works like an inverted pendulum; the wheels are driven in the direction that the upper part of the robot is falling, which means the robot body pitches whenever it moves. Especially when the robot accelerates or decelerates, the pitch angle increases by a significant amount. Since all sensors are directly mounted on the platform, the pitch motions prevent direct image processing. Therefore, the ego-motion compensation step should be able to cope with not only planar movements but also pitch motions.

The *Pioneer2 AT* in Figure 14 (c) is a typical four-wheeled, statically stable robot. Since the Pioneer2 robot is the only statically stable platform among these robot platforms, we drove the robot on the most severe test environment. Figure 17 shows the rocky terrain where the robot was driven. In addition, the moving objects were occluded occasionally because of the trees in the environment.

The computation was performed on embedded computers (Pentium III 1.0 GHz) on the robots. Low resolution (320x240 pixels) input images were chosen for real-time response. The maximum number of particles was set to 5,000, and the minimum number of particles was set to 1000. Since the algorithm is supposed to run in parallel with other processes (eg. navigation and communication), less than 70 percent of the CPU time was dedicated for tracking; the tracking algorithm was able to process five frames per second.

*2) Experimental Results:* The performance of the tracking algorithm was evaluated by comparing with the positions of manually tracked objects. For each video sequence, the rectangular region of moving objects were marked manually and used as ground truth. Figure 15–17 show this evaluation process. The upper rows show the input image sequence, and the positions of manually-tracked objects are marked with rectangles. The lower rows show the set of particles and the clustering results. The position of each particle is marked with dots, and the horizontal bar on the top-left corner of the image indicates the number of particles being used. The clustering result is represented using an ellipsoid and a line inside. The ellipsoid shows the mean and covariance of the estimated object position, and the line inside of the ellipsoid represents the estimated velocity vector of the object.

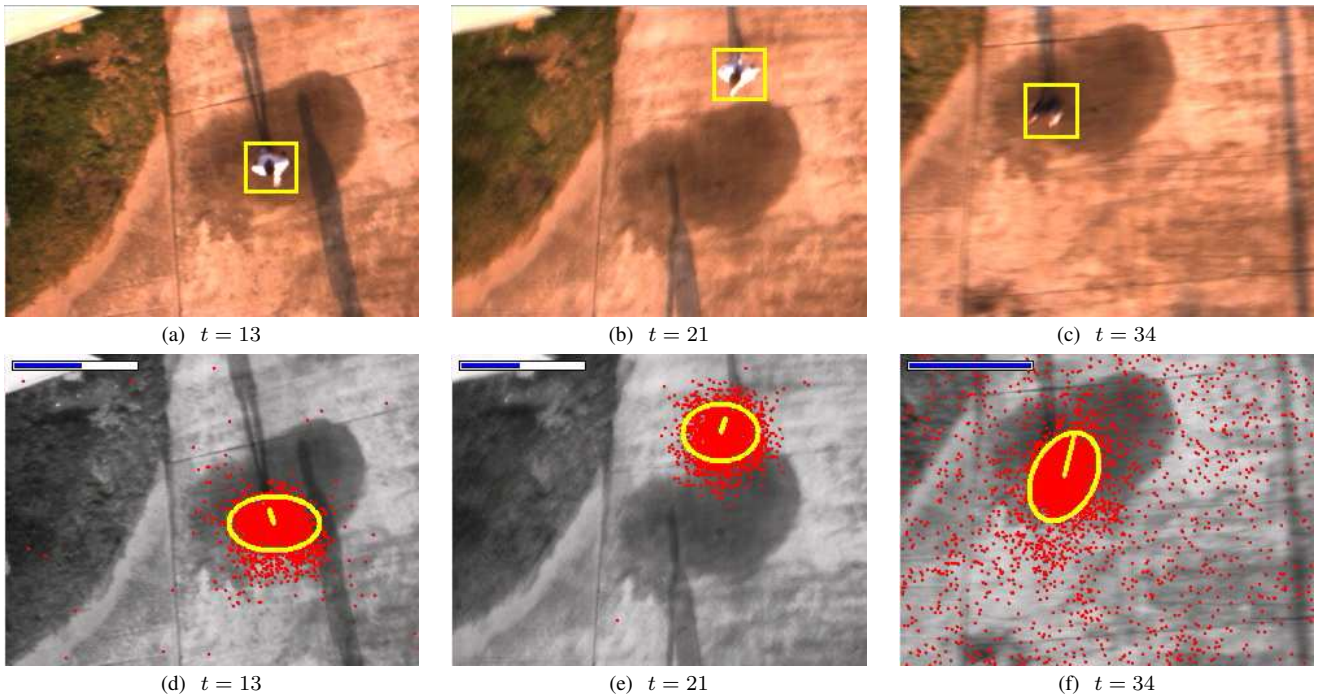
The final evaluation result is shown in Table I. *Frames* is the number of image frames in a video sequence, and *Motions* is the number of moving objects. *Detected* is the total number of detected objects, and *True +* and *False +* are the number of correct detections and the number of false-positives respectively. *Detection Rate* shows the percentage of moving objects correctly detected, and *Avg. Error* is the average Euclidean distance in pixels between the ground truth and the output of tracking algorithm. The average distance error should not be considered as actual error measurement since the tracking algorithm does not perform an explicit object segmentation; it may track a part of an object that generates motion while the ground truth always tracks the whole objects even though only part of the object moves.

The *Robotic helicopter* result shows that the tracking algorithm missed seven objects, but five of them were the cases when a moving object was introduced and showed only partially on the boundary of the image plane. Once the whole object entered into the field of view of the camera, the tracking algorithm tracked it robustly. For the *Segway RMP* result, the detection rate was satisfactory, but the average distance error was larger than the others. The reason was that the walking person was closer to the robot and the tracking algorithm often detected the upper body only, which caused a constant distance error. The *Pioneer2 AT* result shows the higher ratio of false-positives; however, as explained in the previous section, the terrain for the experiment was more challenging (rocky) and the input images were more blurred and unstable. Overall various types of ego-motions were successfully eliminated from input images, and the particle filter was able to track motions robustly from diverse robot platforms.

### B. Tracking Multiple Moving Objects

The multiple-motion tracking system using multiple particle filters was introduced in Section IV-D. Since the robustness of an individual filter was analyzed in Section VI-A, we focus on analyzing how multiple filters are created and destroyed effectively when the number of moving objects changes dynamically.

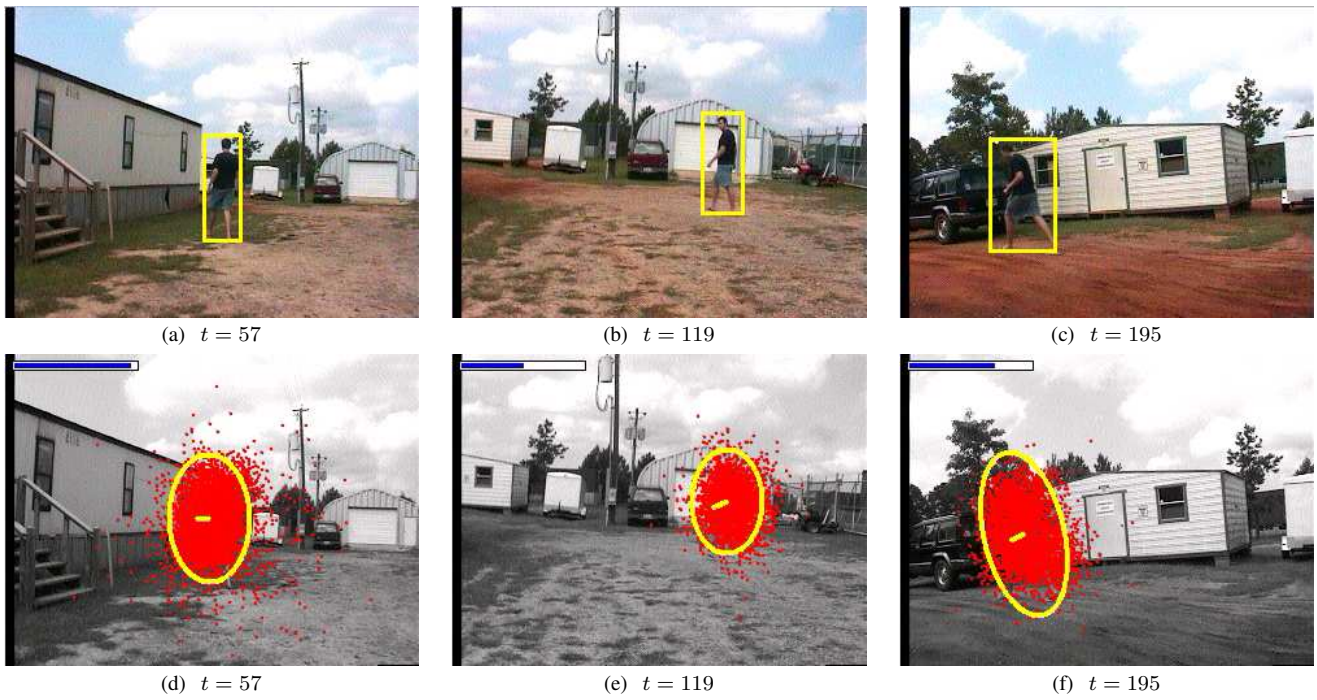
*1) Experimental Setup:* The *Segway RMP* in Figure 14 (b) was selected for the experiment because of its complex ego-motion. The *Segway RMP* is a dynamically stable platform, and its pitching motions for self-balancing are combined into



(a) $t = 13$ (b) $t = 21$ (c) $t = 34$

(d) $t = 13$ (e) $t = 21$ (f) $t = 34$

Fig. 15. Moving object tracking from Robotic helicopter: The upper row shows the input image sequence with manually-tracked objects, and the lower row shows the particle filter outputs and clustering results.



(a) $t = 57$ (b) $t = 119$ (c) $t = 195$

(d) $t = 57$ (e) $t = 119$ (f) $t = 195$

Fig. 16. Moving object tracking from Segway RMP: The upper row shows the input image sequence with manually-tracked objects, and the lower row shows the particle filter outputs and clustering results.

TABLE I

PERFORMANCE OF MOVING OBJECT DETECTION ALGORITHM

| Platform | Frames | Motions | Detected | True + | False + | Detection Rate | Avg. Error |
|---|---|---|---|---|---|---|---|
| Robotic helicopter | 43 | 35 | 28 | 28 | 0 | 80.00 % | 11.90 |
| Segway RMP | 230 | 220 | 215 | 211 | 4 | 95.90 % | 21.31 |
| Pioneer2 AT | 195 | 172 | 158 | 146 | 12 | 84.88 % | 15.87 |

(a) $t = 35$ (b) $t = 56$ (c) $t = 191$

(d) $t = 35$ (e) $t = 56$ (f) $t = 191$

Fig. 17. Moving object tracking from Pioneer2 AT: The upper row shows the input image sequence with manually-tracked objects, and the lower row shows the particle filter outputs and clustering results.

linear and angular motions. The combination of these motions causes complicated ego-motions even when the robot is driven on a flat terrain or when the robot stops in place. The robot was driven on the USC campus during the daytime when there are diverse activities in the environment including walking people and automobiles.

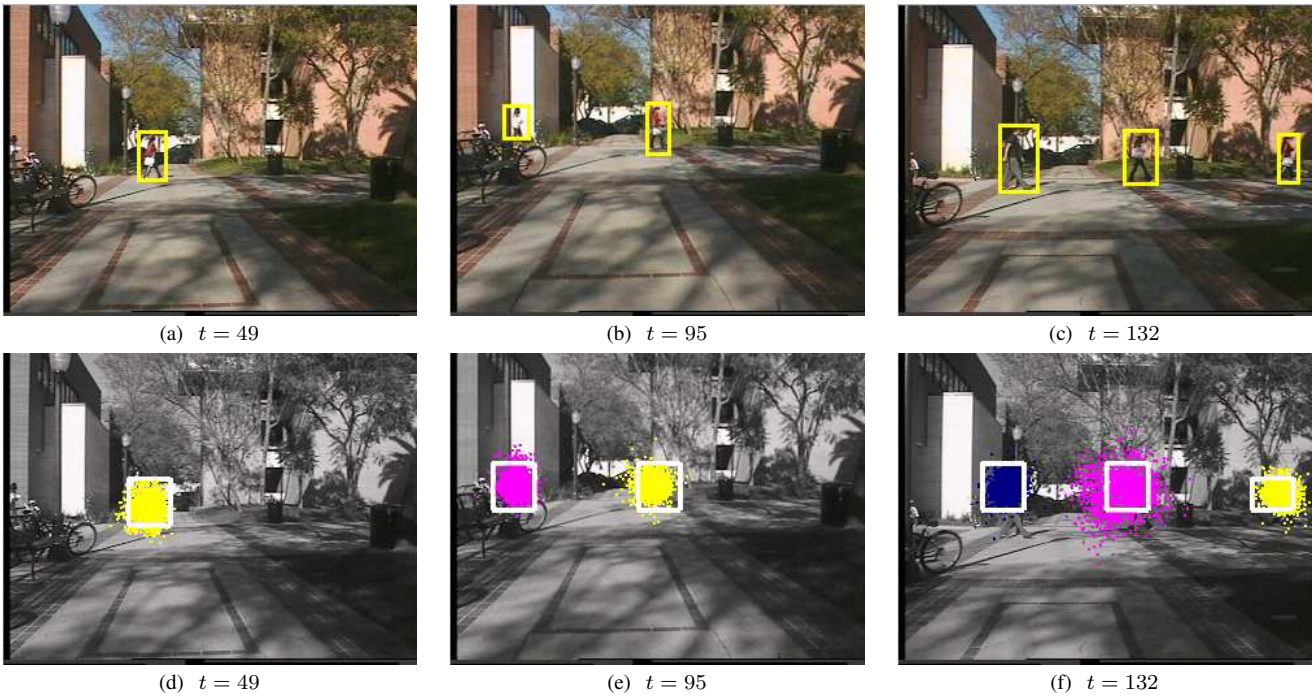
The tracking performance is analyzed for three different cases. As explained in Section IV-D, if one of the following two conditions is not satisfied, a single particle filter fails to track multiple objects even though it supports multi-modality in theory: (1) all objects should be introduced before a particle filter converges, and (2) the convergence speed of a particle filter should be sacrificed by using a *"bad"* perception model. The first two cases are when one or both conditions can not be satisfied. In the first case, there are three people walking by, but the people are introduced in the input image sequence one by one, which violates the first condition. In the second case, there are two groups of automobiles passing by, and they are introduced sequentially with a big time interval between them. In addition, the automobiles move fast enough so that the convergence speed of a particle filter cannot be sacrificed, which violates the second condition. The results for both cases show how the multiple particle filter approach overcomes the limitation of a single particle filter. The stability of this approach is also clear. In the last case, it is observed how multiple particle filters behave when two people walk in different directions and intersect in the middle.

The computation was performed on a Pentium IV (2.1 GHz) computer, and the image resolution was fixed to 320x240 pixels. The maximum number of particle filters was fixed to five, and for an individual particle filter, the range of the number of particles was set to $(1000 \sim 5000)$. The number of frames processed per second varies based on how many particle filters have been created, but roughly 10 frames were able to be processed.

*2) Experimental Results:* The snapshots of the multiple particle filter tracking multiple moving objects are shown in Figure 18–20. The upper rows of the figures show input image sequences and manually-tracked moving objects in the images. The manually-tracked objects are marked with rectangles. The lower rows show particle filters and the covered area (the minimum rectangular region enclosing each ellipsoid that is generated by the particle clustering algorithm) by each particle filter. Only converged particle filter is visualized on the images. Each particle filter is drawn with different colored dots, and the covered areas are marked with rectangles.

The experimental result of the first case is shown in Figure 18. The estimation process starts with a single particle filter. When the first person enters into the field of view of the camera as in Figure 18 (a), the particle filter converges and starts to track the person as in Figure 18 (d), and a new particle filter is created to explore the remained area. When the second person enters as in Figure 18 (b), the new particle filter converges and starts to track the second person as in Figure 18 (e), and another particle filter is created. This process is repeated whenever a new object is introduced. At the end when three people are in the input image as in Figure 18 (c), the total number of particle filters becomes four; three filters for people and one extra filter to explore.

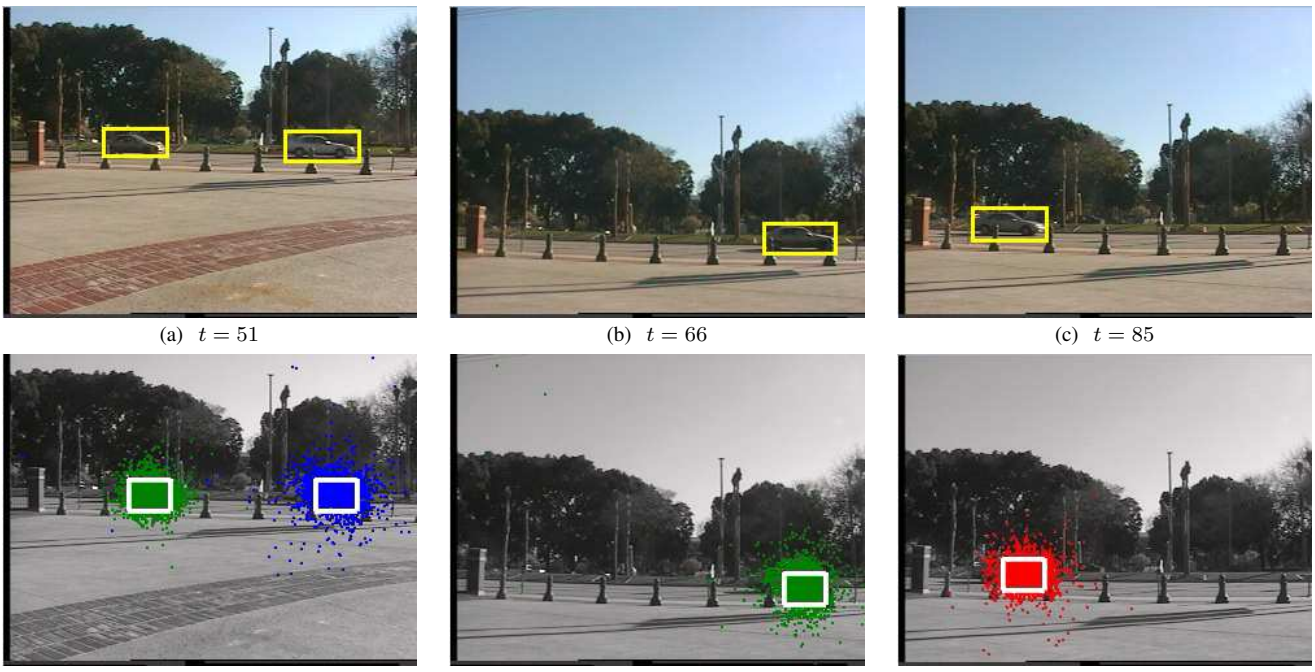
Figure 19 shows the experimental result of the second case. The estimation process is performed in the same way with the first case. Whenever a new automobile is introduced, a new particle filter is created. When the automobile leaves from the field of view of the camera, the particle filter that tracks the



(a) $t = 49$ (b) $t = 95$ (c) $t = 132$

(d) $t = 49$ (e) $t = 95$ (f) $t = 132$

Fig. 18. Tracking people: There are three people walking by. The upper row shows the input image sequence with manually-tracked objects, and the lower row shows the particle filter outputs and clustering results.



(a) $t = 51$ (b) $t = 66$ (c) $t = 85$

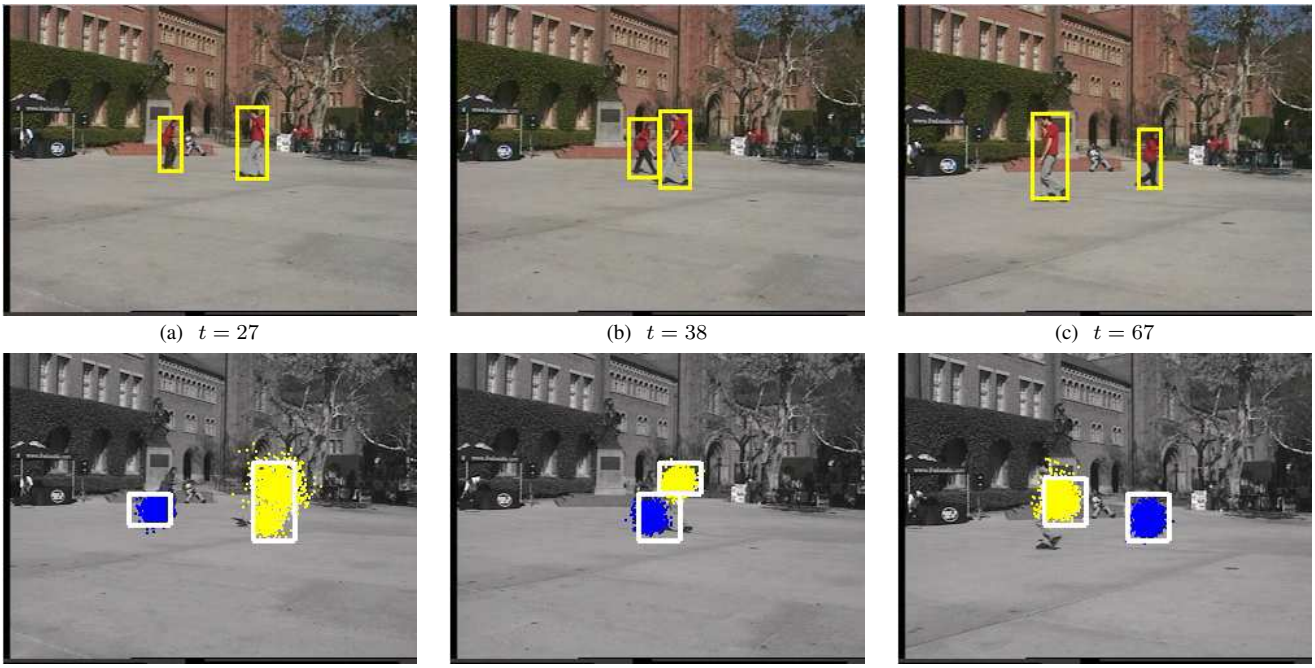
(d) $t = 51$ (e) $t = 66$ (f) $t = 85$

Fig. 19. Tracking automobiles: There are two cars passing by followed by a third car. The upper row shows the input image sequence with manually-tracked objects, and the lower row shows the particle filter outputs and clustering results.

TABLE II

PERFORMANCE OF MOVING OBJECT DETECTION ALGORITHM

| Case | Frames | Motions | Detected | True + | False + | Detection Rate | Avg. Error |
|---|---|---|---|---|---|---|---|
| (1) Pedestrians | 141 | 285 | 274 | 274 | 0 | 96.14 % | 8.68 |
| (2) Automobiles | 123 | 120 | 95 | 92 | 3 | 76.67 % | 20.40 |
| (3) Intersection | 81 | 162 | 156 | 156 | 0 | 96.30 % | 12.34 |



(a) $t = 27$ (b) $t = 38$ (c) $t = 67$

(d) $t = 27$ (e) $t = 38$ (f) $t = 67$

Fig. 20. Intersectional particles: There are two people intersecting in the image. The upper row shows the input image sequence with manually-tracked objects, and the lower row shows the particle filter outputs and clustering results.

automobile diverges and is destroyed eventually.

The experimental result of the third case is shown in Figure 20. There are two people walking in different directions as in Figure 20 (a), and two particle filters are created to track them individually as in Figure 20 (d). The pedestrians intersect in the middle, and keep walking in each direction. Figure 20 (e) and (f) demonstrate that the particle filters track them successfully without being confused by the intersection.

The detailed evaluation result is shown in Table II. There were untracked motions in common; however, it happens only right after a new object is introduced and before a filter converges on the object. Once a particle filter converges and is associated with the object, it never fails to track the object. For the second case, the detection rate is lower than the other two cases. This is reasonable since the motion of an automobile is much faster than that of a person, and it took longer for a particle filter to converge on it. The false-positives observed in the second case are also related to the faster motion. When an automobile leaves from the camera field of view, it disappears quickly enough so that the particle filter stays converged for one or two frames. In general, the multiple particle filter approach shows stable performance for all cases.

### C. Close the loop: Following a Moving Object

The proposed tracking system is integrated with a robot control loop, and its robustness and real-time capability are tested. For this experiment, the task of a robot is to wait for a moving object to appear and follow the object.

*1) System Design and Implementation:* A robot needs two capabilities to accomplish the task: motion detection and tracking, and local navigation. For motion detection and tracking, the system described in Section III-V is utilized. This

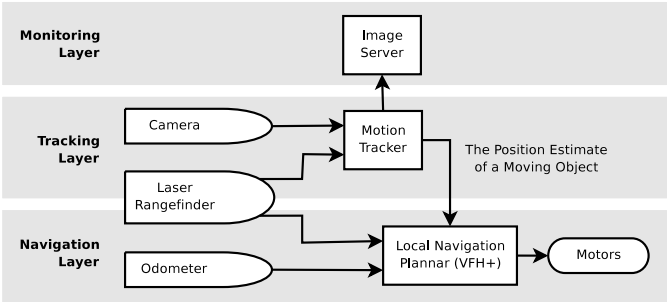



Fig. 21. Control architecture for motion following system: The "motion tracker" box represents the proposed motion tracking system.

module takes the sequence of camera images and the laser range scans as inputs, and computes the existence of moving object(s) and the estimation of target positions in the robot's local coordinates. For local navigation, VFH+ (Vector Field Histogram +) [34] algorithm is implemented. Internally, VFH+ algorithm performs two tasks: (1) it retrieves range scans from a laser rangefinder, and build a local occupancy grid map for obstacle avoidance, and (2) when the estimated target position is given, the algorithm generates both translational and rotational motor commands for point-to-point navigation.

The system architecture is presented in Figure 21. The implemented system was deployed on a Segway RMP robot. The computation was performed on an embedded computer (Athlon 1.0 GHz) on the robot, and the image resolution was fixed at 320x240 pixels.

*2) Experimental Result:* The snapshots of the robot following a person are shown in Figure 22. When the person entered into the field of view of the camera as in Figure 22 (a), the particle filter converged on him, and the Segway started

to follow it. As shown in Figure 22 (b)-(e) and (h)-(i), there were automobiles, a golf car, and pedestrians passing by in the background. However, the tracking system was not confused by them; the Segway was able to track the person without a single failure. When the person stopped and stood still as in Figure 22 (l), the particle filter diverged, which made the robot stop. However, when the person re-started to walk as in Figure 22 (m), the particle filter was able to detect the motion again, and the robot re-started to follow the person.

## VII. Conclusion

We have presented a set of algorithms for multiple motion tracking from a mobile robot in real-time. There are three challenges: *1) Compensation for the robot ego-motion:* The ego-motion of the robot was directly measured using corresponding feature sets in two consecutive images obtained from a camera rigidly attached to the robot. In order to eliminate the unfavorable effect of a moving object in the image sequence, an outlier detection algorithm has been proposed. *2) Transient and structural noise:* An adaptive particle filter has been designed for robust motion tracking. The position and velocity of a moving object were estimated by combining the perception model and the motion model incrementally. Also, the multiple target tracking system has been designed using multiple particle filters. *3) Sensor fusion:* The depth information from a laser rangefinder was projected into the image space, and the partial 3D position information was constructed in the region of overlap between the range data and the image data.

The proposed algorithms have been tested with various configurations in outdoor environments. First, the algorithms were deployed on three different platforms (*Robotic Helicopter*, *Segway RMP*, and *Pioneer2 AT*), and tested in different environments. The experimental results showed that various type of ego-motions were successfully eliminated from input images, and the particle filter was able to track motions robustly. Second, the multiple target tracking algorithm was tested for different types of motions. The experimental results show that multiple particle filters are created and destroyed dynamically to track multiple targets introduced at different times. Lastly, the tracking algorithm was integrated with a robot control loop to test its real-time capability, and the task of following a moving object was successfully accomplished.

The proposed algorithms are expected to be utilized in various application domains as a key enabler. *Localization and mapping* problems have been studied actively by the mobile robotics community, and there are many well-developed techniques widely used. However, most techniques assume a static environment, and their performance is degraded significantly when there are dynamic objects in an environment. Our algorithm provides a robust method to detect dynamic objects in an environment, and it can be exploited in a pre-processing step to filter out data that are associated with the dynamic objects. *Safe navigation* is another fundamental problem in mobile robotics, and most solutions generate motion commands based on local positions of obstacles. However, those solutions become unreliable when obstacles are dynamic, especially when the obstacles move faster than the robot. In this case, a mobile robot needs to predict obstacle position in the near future to avoid collision. The motion velocity (speed and direction) estimation capability of our algorithm could fulfill this requirement. *Human-robot interaction* is active research area in service robot applications, and locating a subject to interact with is a key problem. Our algorithm is applicable in this regard. Needless to say, *surveillance and security* applications could make use of our algorithms to detect and track moving targets.

## Acknowledgment

This work is supported in part by DARPA grants DABT63-99-1-0015, and 5-39509-A (via UPenn) under the Mobile Autonomous Robot Software (MARS) program, DOE RIM under grant DE-FG03-01ER45905, and NSF CAREER grant IIS-0133947.

## References

[1] D. Wolf and G. S. Sukhatme, "Online simultaneous localization and mapping in dynamic environments," in *Proceedings of the International Conference on Robotics and Automation*, New Orleans, Louisiana, April 2004, pp. 1301–1306.

[2] D. Schultz, W. Burgard, D. Fox, and A. B. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 2001, pp. 1165–1170.

[3] M. Montemerlo, S. Thrun, and W. Whittaker, "Conditional particle filters for simultaneous mobile robot localization and people-tracking," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington DC, May 2002, pp. 695–701.

[4] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-91-132, April 1991.

[5] A. Censi, A. Fusiello, and V. Roberto, "Image stabilization by features tracking," in *Proceedings of the 10th International Conference on Image Analysis and Processing*, Venice, Italy, September 1999, pp. 665–667.

[6] I. Zoghlami, O. Faugeras, and R. Deriche, "Using geometric corners to build a 2D mosaic from a set of images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 420–425.

[7] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981, pp. 674–697.

[8] S. Srinivasan and R. Chellappa, "Image stabilization and mosaicking using the overlapped basis optical flow field," in *Proceedings of IEEE International Conference on Image Processing*, October 1997, pp. 356–359.

[9] M. Irani, R. Rousso, and S. Peleg, "Recovery of ego-motion using image stabilization," in *Proceedings of the IEEE Computer Vision and Pattern Recognition*, March 1994, pp. 454–460.

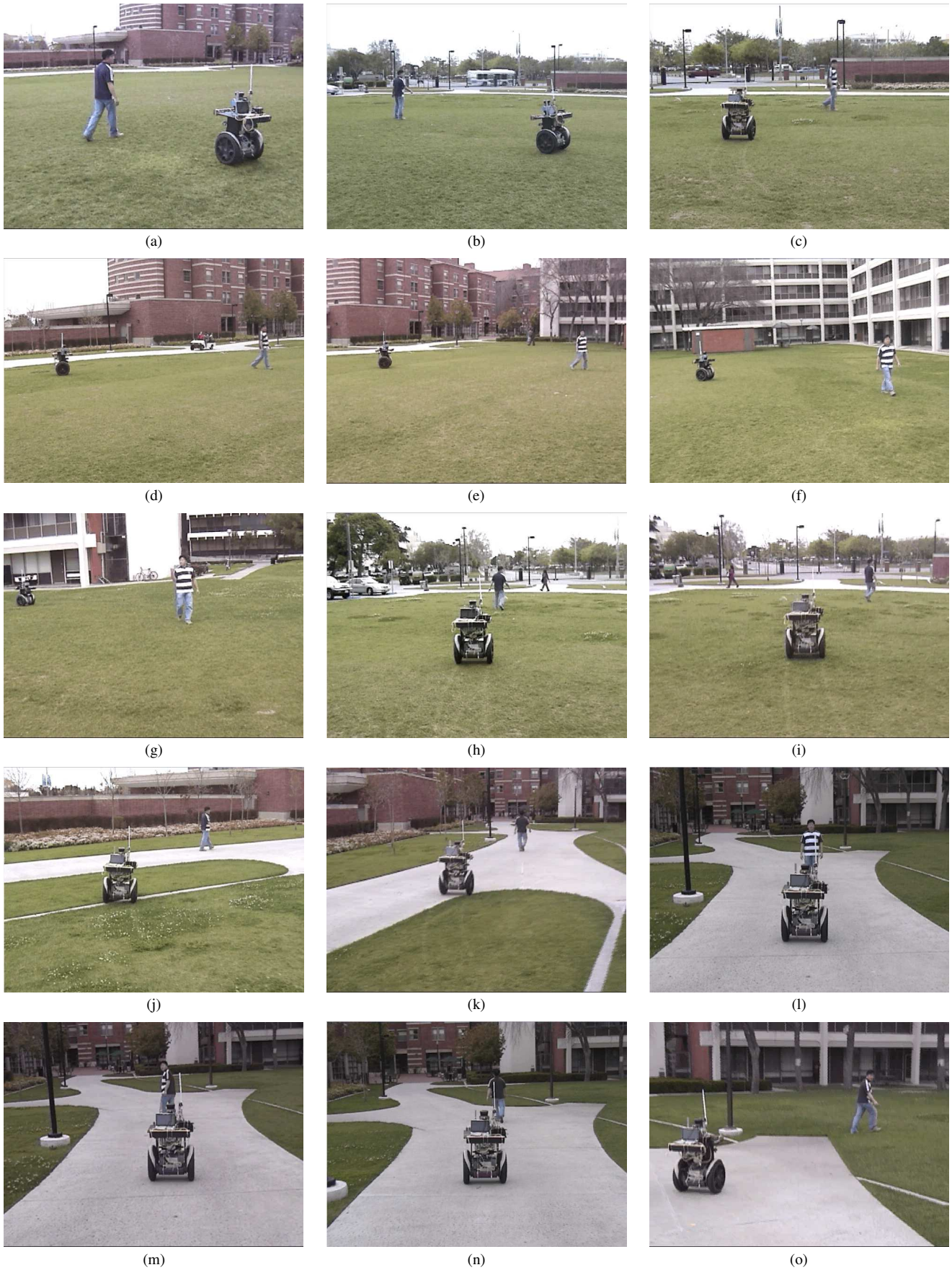
[10] P. Nordlund and T. Uhlin, "Closing the loop: Detection and pursuit of a moving object by a moving observer," *Image and Vision Computing*, vol. 14, pp. 265–275, May 1996.

[11] D. Murray and A. Basu, "Motion tracking with an active camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 449–459, May 1994.

[12] G. L. Foresti and C. Micheloni, "A robust feature tracker for active surveillance of outdoor scenes," *Electronic Letters on Computer Vision and Image Analysis*, vol. 1, no. 1, pp. 21–34, 2003.

[13] A. Yilmaz, K. Shafique, N. Lobo, X. Li, T. Olson, and M. a. Shah, "Target-tracking in FLIR imagery using mean-shift and global motion compensation," in *Workshop on Computer Vision Beyond the Visible Spectrum*, Kauai, Hawaii, December 2001, pp. 54–58.

[14] A. Behrad, A. Shahrokni, and S. A. Motamedi, "A robust vision-based moving target detection and tracking system," in *the Proceeding of Image and Vision Computing Conference*, University of Otago, Dunedin, New Zealand, November 2001.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n) (o)

Fig. 22. Snapshots of a Segway RMP robot following a person

[15] M. B. van Leeuwen and F. C. Groen, "Motion interpretation for in-car vision systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL, Lausanne, Switzerland, October 2002, pp. 135–140.
[16] C. Hue, J.-P. L. Cadre, and P. Pérez, "A particle filter to track multiple objects," in *IEEE Workshop on Multi-Object Tracking*, Vancouver, Canada, July 2001, pp. 61–68.
[17] J. Kang, I. Cohen, and G. Medioni, "Continuous multi-views tracking using tensor voting," in *Proceedings of the IEEE Workshop on Motion and Video Computing*, Orlando, Florida, December 2002, pp. 181–186.
[18] A. Kelly, "A 3D state space formulation of a navigation Kalman filter for autonomous vehicles," The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, Tech. Rep. CMU-RI-TR-94-19, May 1994.
[19] S. I. Roumeliotis and G. A. Bekey, "3D localization for a mars rover prototype," in *5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, ESTEC, Noordwijk, The Netherlands, June 1999, pp. 441–448.
[20] S. Saripalli, J. M. Roberts, P. I. Corke, G. Buskey, and G. S. Sukhatme, "A tale of two helicopters," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2003, pp. 805–810.
[21] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2003.
[22] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *Journal of Intelligent and Robotic Systems*, vol. 18, pp. 249–275, 1997.
[23] J.-S. Gutmann and C. Schlegel, "Amos: Comparison of scan matching approaches for self-localization in indoor environments," in *Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*, 1996, pp. 61–67.
[24] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, Washington, June 1994, pp. 593–600.
[25] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker: Description of the algorithm," Intel Research Laboratory, Tech. Rep., 1999.
[26] G. Welch and G. Bishop, "An introduction to the Kalman filter," Department of Computer Science, University of North Carolina at Chapel Hill, Tech. Rep. 95-041.
[27] J. Kang, I. Cohen, and G. Medioni, "Continuous tracking within and across camera streams," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin, June 2003, pp. 267–272.
[28] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.
[29] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, pp. 99–141, 2001.
[30] D. Fox, "KLD-sampling: Adaptive particle filter," in *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.
[31] B. Jung and G. S. Sukhatme, "Detecting moving objects using a single camera on a mobile robot in an outdoor environment," in *International Conference on Intelligent Autonomous Systems*, The Netherlands, March 2004, pp. 980–987.
[32] S. Thrun, W. Burgard, and D. Fox, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning and Autonomous Robots (joint issue)*, vol. 31 & 5, pp. 29–53 & 253–271, 1998.
[33] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Visually-guided landing of an unmanned aerial vehicle," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 371–381, June 2003.
[34] I. Ulrich and J. Borenstein, "VFH+: Reliable obstacle avoidance for fast mobile robots," in *Proceeding of the IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 16–21 1998, pp. 1572–1577.